Enhancing a Shared-Access, Hardware-Based, Random Number Generation System

Operating Systems FUSE File-System Development Support

James Wolfer(✉)

Indiana University South Bend, South Bend, USA
jwolfer@iusb.edu

Abstract. High quality random numbers form a critical foundation for computing in applications such as data encryption, simulation, and modeling. Recognizing the import of random numbers we have integrated hardware-based random bit generation into a major file system project for the Operating Systems class. Originally built around background radiation events detected by a Geiger counter, we are in the process of extending this to additional hardware-based random number generators configured for shared access by student teams. This work-in-progress documents the most recent deployment of this technology.

Keywords: Hardware random numbers · File systems · Operating systems Pedagogy

1 Introduction

1.1 Overview

High quality random numbers form the basis for much of our modern security infrastructure, in particular data encryption and secure transmission. In addition, many simulation models rely on quality random numbers for accuracy. There are a variety of physically based random number generation approaches such as those surveyed in [1]. As part of an active-learning approach [2] we have previously reported on using background radiation as detected by a Geiger counter as the basis for random numbers to support file-systems projects in the Operating Systems class [3]. This work describes an extension of that project providing additional hardware-based random number resources configured for shared, remote access and its initial deployment.

1.2 The Operating Systems Class

The Operating Systems class at our university integrates operating systems principles with a variety of hands-on projects ranging from probing internal inter-process communications, re-writing scheduling algorithms, memory management, and file systems. Operating Systems is a designated senior-level capstone course in our Computer Science

M. E. Auer et al. (eds.), *Teaching and Learning in a Digital World*,
Advances in Intelligent Systems and Computing 716,
https://doi.org/10.1007/978-3-319-73204-6_78

program, and a variety of experiences ranging from programming to collaborative work is expected as an attempt to consolidate student knowledge and experience. Overt attempts to make these projects as real-life as possible are encouraged since students typically take this course just prior to graduation and joining the workforce.

One of these projects is to implement, under the Linux operating system, a FUSE (File System in User Space) files system to expose hardware-based random bits to the user. This project typically takes four to five weeks, and includes regular feedback as well as semi-flipped classroom activity.

2 The Hardware Infrastructure

2.1 Existing Hardware

Figure 1 shows the original hardware interface consisting of a Geiger Counter interfaced into a cluster of four Raspberry PI computers through the GPIO interface. Background radiation events generate pulses on an interface pin in each computer which are detected and distributed to student programs through a callback interface as detailed in [3]. Students then use the relative time intervals between radiation events to generate random bits. Student home directories are NFS (Network File System) mounted so that each computer has all of the student's files available upon login. With the current generation of four-core Raspberry Pi computers we can support up to six teams of five students each on two devices without issue.

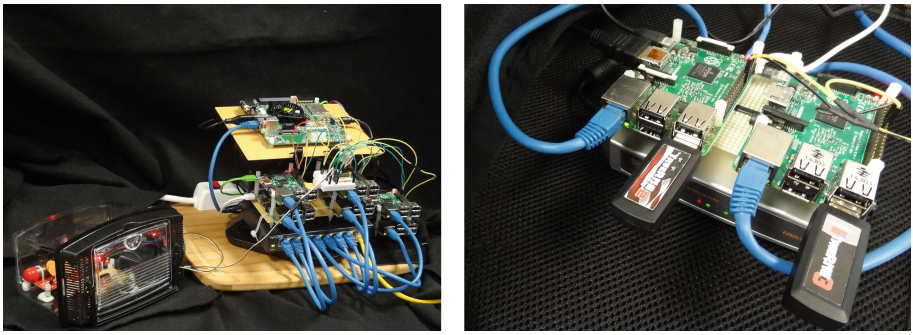

Fig. 1. Geiger counter (left) and TrueRNG-2 and -3 (right) random number hardware

While the Geiger counter based system has proven effective it is a relatively slow interface, typically yielding about ten to fifteen random bits per minute. This requires students to "harvest" bits 24/7 to accumulate enough to feed their files system users. Furthermore, having additional approaches to random number generation allows for comparative assignments.

2.2 Enhanced System

The system was enhanced by adding two USB hardware-based commercial random number generators shown in Fig. 1. Specifically, the Ubld.iT TrueRNG-2 and TrueRNG-3 [4]. Ubld.iT targets applications such as security such as SSH (Secure Shell), gaming programs such as dice, lottery, etc., statistical sampling and simulation as targets for their technology. In our application these devices plug into the USB interface, one attached to each Raspberry PI, and are made available as read-only devices to the students. Again, this is in addition to the Geiger counter system interfaced through the GPIO pins of each computer.

The TrueRNG systems derive their random bits by exploiting the avalanche effect across biased semiconductor junctions. Specifically, the semiconductor junction is biased to 12 volts then digitized. The resulting data is statistically whitened proor to being made available on the USB interface. The interface presents to the system as a serial device, and is capable of 350 kilobits/second for the TrueRNG-2, and 400 kilobits/second for the TrueRNG-3.

There are minor differences between the two models which we anticipate exploiting in future assignments. The TrueRNG-2 exhibits, according to the manufacture, a "very minor" bias when averaged over 100's of megabytes of random data. While unlikely to cause simulation or cryptographic concerns, it is statistically detectable. The TrueRng-3 device reduces this bias to below detectable levels. Having a sample of each model allows for comparative assignments to evaluate the quality of each unit as well as that of the Geiger counter output.

2.3 Additional Capabilities

To illustrate the output of the various hardware generators we enlist a combination of scientific visualization and statistical probes from the Fourmilab "Ent," or entropy, program. The "Ent" program produces a series of statistics over the random bits [5].

We have used a variety of visualizations for this project. For example, Fig. 2 shows the output of a real-time visualization from the Geiger counter interface in which the location and color of the squares are selected using hardware random numbers. Figure 3 exhibits the random bits as pixels in a grayscale image. Note the visual uniformity. Had there been a substantial bias it would be revealed as noticeable lines or clusters in the display.

Figure 4 shows the same information for the TrueRNG-2 and TrueRNG-3 devices. Again note the uniform texture exhibited by the random number distribution from these devices.

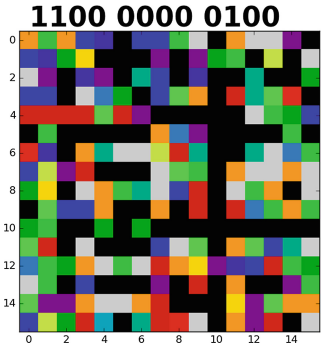

Fig. 2. Geiger counter random number visualization

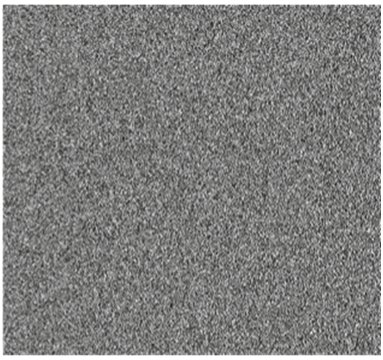

Fig. 3. Geiger counter random image

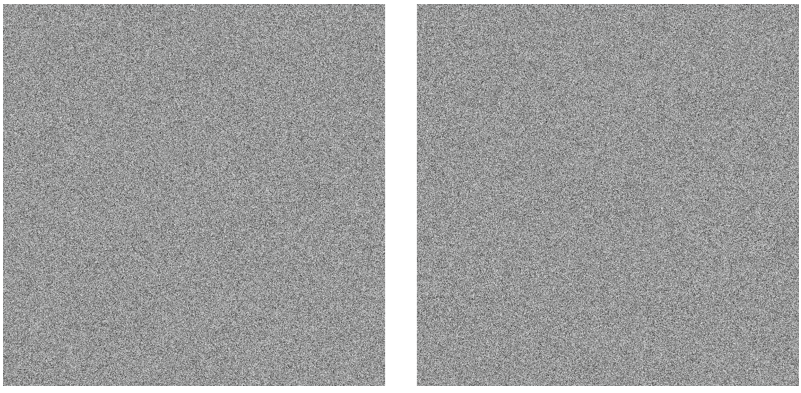

Fig. 4. TrueRNG-2 (right) and TrueRng-3 (left) random images

Table 1 shows the result of running the "Ent" program, which provides an indication of the quality of the random sequence from the Geiger counter pre- and post-bias adjusted as well as those from the TrueRNG-2 and TrueRNG-3 devices. The statistics

include entropy, or the information density of the file. In this case the entropy is 1.0 bits/bit for all cases. Chi-square, which is particularly sensitive to sequence errors, and the arithmetic mean should approach 0.5. Pi is estimated using randomly generated points intersecting a circle inscribed in a square, and the serial correlation coefficient measures the codependence of adjacent samples.

Table 1. ENT random test

	Geiger counter pre-bias removal	Geiger counter post-bias removal	TrueRNG-2	TrueRNG-3
Entropy	1.0	1.0	1.0	1.0
Chi-square	0.671601	0.328496	0.53	0.08
Mean	0.499635	0.50008	0.4999	0.5
Monte-Carlo-Pi	3.458389	3.137899	3.141209	3.139652
Serial-correlation	−0.333256	−0.000454	−0.00016	−0.00004

The ability to compare devices opens the horizon for analytical assignments in the Operating Systems class. While the "Ent" program produces preliminary assessments there are more sophisticated random number test suites, such as those from NIST [6], allowing students to perform deeper analytics for deeper insight.

3 Observations and Future Work

The hardware enhancements described here were deployed late in the last semester. At that point students were already working on the Geiger counter FUSE system, and use of the new hardware was not mandatory. That said, students did try the system and informal feedback indicates that they appreciated having the faster number generator for debugging purposes, something not considered during the build out of the system. In general students expressed enthusiasm for hardware exposure and the FUSE file system project. While they find it challenging, they also recognize that it approaches a "real world" problem at an opportune time in their program.

We believe that this is the beginning of a more comprehensive hardware random number generation suite. Future plans include enhancing the devices with multiple additional hardware approaches. Under active consideration are exposing the built-in ARM random number generator on the Raspberry PI, installing the Free Software Foundation's NeuG [7] random number generator, and exploring the addition of Software Defined Radio for atmospheric noise based random numbers. We also anticipate a suite of software projects to compare the various hardware approaches.

4 Conclusion

The enhancement to our hardware random number generation platform expands the range of assignments appropriate for our Operating Systems students. It also stands to

accelerate their development process. The resulting system forms a solid basis for future projects and holds potential for increasing awareness of the role of random numbers in their programming careers, gives them an appreciation for the limits of pseudo-random numbers, and provides experience delivering those numbers through the operating system. We believe that continuing to build out this platform enhances their intellectual horizon.

References

1. Thomas, A.A., Paul, V.: Random number generation methods a survey. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **6**(1), 556–559 (2016)
2. Arbelaiz, O., Martin, L.I., Muguerza, J.: Analysis of introducing active learning methodologies in a basic computer architecture course. IEEE Trans. Educ. **58**(2), 110–116 (2015)
3. Wolfer, J., Keeler, W.: From Geiger-counters to file systems: remote hardware access for the operating systems course. Int. J. Online Eng. **8**, 26–31 (2016)
4. Ubld.iT: TrueRNG. http://ubld.it/products/truerng-hardware-random-number-generator/
5. Walker, J.: ENT http://www.fourmilab.ch/random/
6. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. US National Institute of Standards and Technology Special Publication 800-22 revision 1a (2010)
7. Free Software Foundataion: NeuG Random Number Generator. https://shop.fsf.org/storage-devices/neug-usb-true-random-number-generator